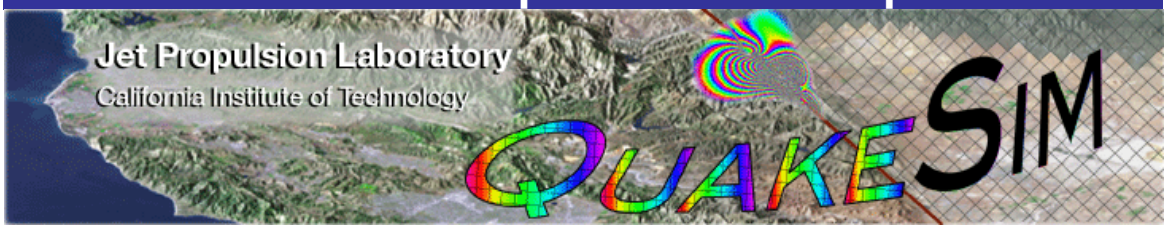
Second Code Improvement Completed



Numerical Simulations For Active Tectonic Processes: Increasing Interoperability And Performance

JPL Task Order: 10650

PARK Documentation Only

Milestone G – Code Improvement due date: 6/30/2004

2nd code improvement - further optimization for some codes, pick up others that were neglected in 1st improvement - documented source code made publicly available via the Web.

- PARK on 1024 CPU machine with 400,000 elements, 50,000 time steps in 5 times the baseline code
- GeoFEST (assuming availability of 880 processor machine) 16M elements, 1000 time steps in the same time as the baseline code using the PYRAMID AMR libraries
- Virtual California with N=700 segments for 10,000 time steps in 1 hour or less, MPI parallel implementation, running on M-processor machine, with 2 GB of memory per CPU, speedup of approximately M/2 on up to 256 processors. Investigation of fast multipole method for this code.

Team

Andrea Donnellan:
Principal Investigator
Jet Propulsion Laboratory
Mail Stop 183-335
4800 Oak Grove Drive
Pasadena, CA 91109-8099
donnellan@jpl.nasa.gov
818-354-4737

Michele Judd:
Technical Task Manager
Jet Propulsion Laboratory
Mail Stop 183-335
4800 Oak Grove Drive
Pasadena, CA 91109-8099
michele.judd@jpl.nasa.gov
818-354-4994

Jay Parker:
Overall Software Engineer
Jet Propulsion Laboratory
Mail Stop 238-600
4800 Oak Grove Drive
Pasadena, CA 91109-8099
Jay.W.Parker@jpl.nasa.gov
818-354-6790

Terry Tullis:
Fast Multipole Methods
Brown University
Box 1846, Brown University
Providence, RI 02912-1846
Terry_Tullis@Brown.edu
401-863-3829

Geoffrey Fox:
Information Architect
Community Grid Computing Laboratory
Indiana University
501 N. Morton, Suite 224
Bloomington, IN 47404-3730
gcf@indiana.edu
812-856-7977

Dennis McLeod:
Database Interoperability
Professor
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
mcleod@usc.edu
213-740-4504

John Rundle:
Pattern Recognizers
Center for Computational Science and Engineering
U. C. Davis
Davis, CA 95616
rundle@geology.ucdavis.edu
530-752-6416

Gleb Morein:
Pattern Recognizers
Center for Computational Science and Engineering
U. C. Davis
Davis, CA 95616
gleb@cse.ucdavis.edu

Greg Lyzenga:
Finite Element Models
Jet Propulsion Laboratory
Mail Stop 126-347
4800 Oak Grove Drive
Pasadena, CA 91109-8099
greg.lyzenga@jpl.nasa.gov
818-354-6920

Marlon Pierce:
Code Interoperability Software Engineer
Community Grid Computing Lab
Indiana University
501 N. Morton, Suite 224
Bloomington, IN 47404-3730
marpierc@indiana.edu
812-856-1212

Lisa Grant:
Fault Database Architect
University of California, Irvine
Environmental Analysis and Design
Irvine, CA 92697-7070
lgrant@uci.edu
949-824-5491

Robert Granat:
Pattern Recognizers
Jet Propulsion Laboratory
Mail Stop 126-347
4800 Oak Grove Drive
Pasadena, CA 91109-8099
robert.granat@jpl.nasa.gov
818-393-5353

Maggi Glasscoe:
GeoFEST Code Verification
Jet Propulsion Laboratory
Mail Stop 300-233
4800 Oak Grove Drive
Pasadena, CA 91109-8099
Margaret.T.Glasscoe@jpl.nasa.gov
818-393-4834

AD HOC Team Member
Charles Norton:
PYRAMID/GeoFEST
Jet Propulsion Laboratory
Mail Stop 169-315
4800 Oak Grove Drive
Pasadena, CA 91109-8099
Charles.Norton@jpl.nasa.gov
818-393-3920

Overview

This milestone report documents the completion of the PARK portion of the Milestone G of the QuakeSim project - *Numerical Simulations for Active Tectonic Processes: Increasing Interoperability and Performance* - for NASA's Earth-Sun System Technology Office, Computational Technology Program. The text of the milestone appears on Page 1 of this report, and requires demonstration of substantially larger problems than Milestones E and F (see Table 1 below).

Code	Machine Wallclock Time	Processors	Date	Elements	Time Steps
PARK Milestone **E**	*Chapman (AMES)* 7.888 Hours	1	September 18, 2002	15,000	500
PARK Milestone **F**	*Chapman* (AMES) 7.879 Hours	256	August 15, 2003	150,000	5,000
PARK Milestone **G**	*Dell Linux Cluster (JPL)* 34.518 Hours*	512	May 27, 2005	400,000	50,000

Table 1: Computer runs demonstrating baseline, Milestone F performance enhancements and Milestone G performance enhancements for PARK. Note that Milestone G wallclock time beat the allowable time of 39.440 hours (five times the baseline run).

Milestone G Supporting Documents

The top-level web site for the QuakeSim task is at http://quakesim.jpl.nasa.gov. Source code for the three codes may be found at http://quakesim.jpl.nasa.gov/download.html. Files required for the baseline cases may be found at http://quakesim.jpl.nasa.gov/milestones.html.

The PARK Code

Problem Being Solved

Compute the history of slip, slip velocity, and stress on a vertical strike-slip fault that results from using state-of-the-art rate and state frictional constitutive laws on the fault for a specific geographic setting at Parkfield, California.

The boundary conditions are those appropriate for Parkfield and the distribution of constitutive properties on the fault zone are as realistic as our ability to characterize the subsurface properties of the fault there allow. The methods developed in solving this problem can be generalized to other geologic settings in which the fault geometry, the boundary conditions are not so simple and multiple faults are involved.

The grid used in Milestone G has 400,000 elements. The primarily change from the 150,000 element model used for Milestone F was a large increase in the number of square elements that are 7.4 meters on an edge, there being 273,375 of these in the current model and only 5040 in the previous model. The increase in the number of these fine elements was made in the area of the model where the earthquake is expected to nucleate. Some minor adjustments in the number of some of the larger elements was made to result in the change from 150,000 to 400,000 elements.

PARK Documentation

The main program is a boundary element program that determines the stress on every element of the fault surface due to slip on every other element, using a Greens function approach. The fault constitutive law is used to determine what the slip velocity will be for that stress and this velocity multiplied by the time step gives the slip to be used to calculate the stress in the next time increment. This involves the forward time integration of coupled ordinary differential equations. The integration for previous milestones was done with a fifth order Runge-Kutta[1] scheme with adaptive step size control.

Because the time-steps range over ten orders of magnitude, depending on whether the fault is slipping very slowly in the interseismic period or very fast during an earthquake, the adaptive step-size control is an essential element in the solution.

For this milestone the fifth order Rung-Kutta routine that involved six stages, namely the determination of derivatives six times for every time step, was replaced by a faster second order routine, that involves only two stages and reuses the derivatives calculated at the end of the previous time step to begin the integration at the beginning of the current time step. The adaptive step size scheme is retained. Tests to date show that, as expected, this second order routine is 3 times faster per time step than the fifth order one. The disadvantage is that each time step is smaller, but tests to date suggest that a given model time is reached in essentially the same CPU time as with the fifth order routine. For future users who may want to also try the fifth order routine that uses two copyrighted subroutines from Numerical Recipes, an alternative version of the

[1] Runge-Kutta is a method for forward integration of differential equations that involves calculating derivatives of the functions at the current time and several fractions of potential time-steps in the future, appropriately weighting these derivatives estimating the best derivative value to use and determining the value of the function at the new time by multiplying that best derivative by the appropriate time-step. The fifth-order Runge Kutta method compares estimates made using two different time-steps and, based on this comparison, determines whether a smaller or larger time-step should be used for the next step. This allows for adaptive time-stepping which is extremely important in problems such as this where the time-steps can vary as much as ten orders of magnitude, depending on whether interseismic or a coseismic behavior is involved.

park.f program is provided that works with those subroutines and contains instructions on how to obtain and modify them to work in this application.

The main program calls a variety of subroutines and the one of these subroutines that calculates the derivatives used in the forward time integration itself calls a Fast Multipole library that is suitable for such Green's functions problems. The Multipole approach allows a number of computations to scale as N log N rather than N^2 as would otherwise be the case. The particular Fast Multipole approach being used allows determination of the degree of grouping of the remote cells based on an analytical approximation to the Greens function. In order to reduce computation time it also renumbers the elements so that those that are near in space are also near in memory.

The main program and most of its subroutines are written in Fortran 90. The Fast Multipole library and its interface program to the main program and its subroutines are written in C. All of these programs run in parallel using MPI.

PARK Scaling Analysis

In the directory **scaling**, found in the same **2nd_Code_Improv_Milestone** directory in which this file is found, are files that show how the job scales with number of processors. Sixty six scaling runs were done, on several different machines, including the Compaq-HP AlphaServer named *halem* at the NCCS at Goddard, the DEC *altix* machines, *Columbia2* and *Columbia17* at NASA Ames, and the Dell Linux cluster at JPL. An attempt was made to do runs for 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 processors in as many of those machines as possible, but this was only possible on the JPL Dell cluster, 256 CPUs being the maximum number tested on *halem* and *columbia2* and 512 being the maximum number on *colubia17*. The scaling tests were done on models with 150,000 elements on *halem* and *columbia2*, and 400,000 on *Columbia17* and the JPL cluster. The number of time steps used in these scaling runs varied from 17 to 100, depending on the machine and time available for the tests, some of which were done with the entire machine dedicated to the tests, whereas the full 400,000 element, Second Code Improvement Milestone run was done for 50,000 time steps. In the scaling directory is a data table giving the time per step for all the scaling runs, as well as plots showing dependence of time per step, efficiency, and overhead on number of processors.

The scaling data show that not much speedup is gained by going from one to two processors, a problem that existed in the scaling in the First Code Improvement Milestone that we have not been able to solve, although extensive efforts were made by the scientists working on this code and by several experts at all three of the sites where the code was run. Whether this problem could be fixed in the future is not clear. It may be an intrinsic problem in the Fast Multipole Library, although for the astrophysics application the library was originally created for this

problem has not been seen. Some continued work on this problem may be warranted, since solving it would decrease run times by a factor of two.

Efficiency and overhead are nearly constant from 2-64 processors on all machines tested, but it falls off at the largest number of processors for all machines tested, due to too much time spent in interprocess communication. The behavior is most easily seen on the overhead plots, as well as by examination of the tables. For *halem* it rises somewhat going to 128 CPUs and in going to 256 CPUs it increases drastically, the time to run on 256 being greater than on 128. For *Columbia2* and *Columbia17* the scaling is good to 256 CPUs but on Columbia17 for which a 512 CPU run was done, it increases drastically for 512 CPUs, taking more time than on 256. Similar behavior was found on the JPL Dell cluster, in which the time on 512 and 256 CPUs was the same. The behavior on 1024 CPUs was even worse, it taking considerably more time than on 512 CPUs. As a result of this behavior, the 50,000 step Second Code Improvement run was actually done on 512 processors on the JPL Dell cluster. The milestone could equally well have been met on 256 CPUs. Due to the poor scaling above 256 CPUs on all machines tested, the only way the milestone was able to be met was to combine the advantages of 1) using the second order Runge Kutta integration scheme and 2) running on the fastest of all the machines tested, namely the JPL Dell cluster. The relative speed of the different machines can be see both in the tables and in the plot of time per step vs. number of CPUs.

The implications of these scaling runs is that it is important to do scaling tests on any new machine the code may be ported to and, with the results of that available, to run it on a small enough number of processors that the inter-process communication does not dominate the behavior. With their current hardware, on Columbia and the JPL cluster the largest number of processors that should be used is 256. The usable number of processors might increase if many more than 400,000 elements were used, but the behavior is similar for both 150,000 and 400,000 elements.

PARK Walltime/Time Step (hours)							
Machine No. Elements CPUs	JPL Dell Cluster 400,000	Columbia17/15 400,000	Columbia8 400,000	Columbia8 150,000	Halem 150,000	Halem * 150,000	Chapman 150,000
1	0.08181	0.24895	0.39817	0.15203	0.06953	0.06189	0.07963
2	0.07658	0.23382	0.37647	0.14384	0.06873	0.06023	0.07699
4	0.04743	0.14639	0.24136	0.07318	0.03270	0.02883	0.04039
8	0.02232	0.06639	0.11003	0.03783	0.01857	0.01626	0.02027
16	0.01234	0.03642	0.05995	0.02422	0.01221	0.00836	0.01290
32	0.00665	0.02014	0.03266	0.01259	0.00633	0.00551	0.00644
64	0.00358	0.01105	0.01739	0.00668	0.00426	0.00354	0.00347
128	0.00206	0.00719	0.01015	0.00388	0.00413	0.00312	0.00203
256	0.00145	0.00383	0.00598	0.00251		0.00369	0.00158
**256	0.00127						
512	0.00128	0.00905					
1024	0.00229						

Table 2: The data shown here are for the 6-stage, fifth order Runge Kutta routine and so the times are 3 times longer than would be the case if the 2-stage second order routine were used as was the case for the 50,000 time step milestone run. Blank entries indicate no run done for that machine on that number of CPUs. *The runs on *halem* in this column were done with incorrect keys, so the timing relative to the other correct runs is not useful, but it is the only 256 CPU run done on *halem* and consequently it is included since the scaling data is perfectly valid. The data for this column are shown in the Efficiency and Overhead plot, but not in the walltime plot. ** Only 17 time steps were used for the rows above this one on the JPL Dell, but the data in this and the next two rows were based on 100 times steps. The larger number of time steps gives more accurate data, but the difference is small as shown in the accompanying plot and by comparing this and the row above.

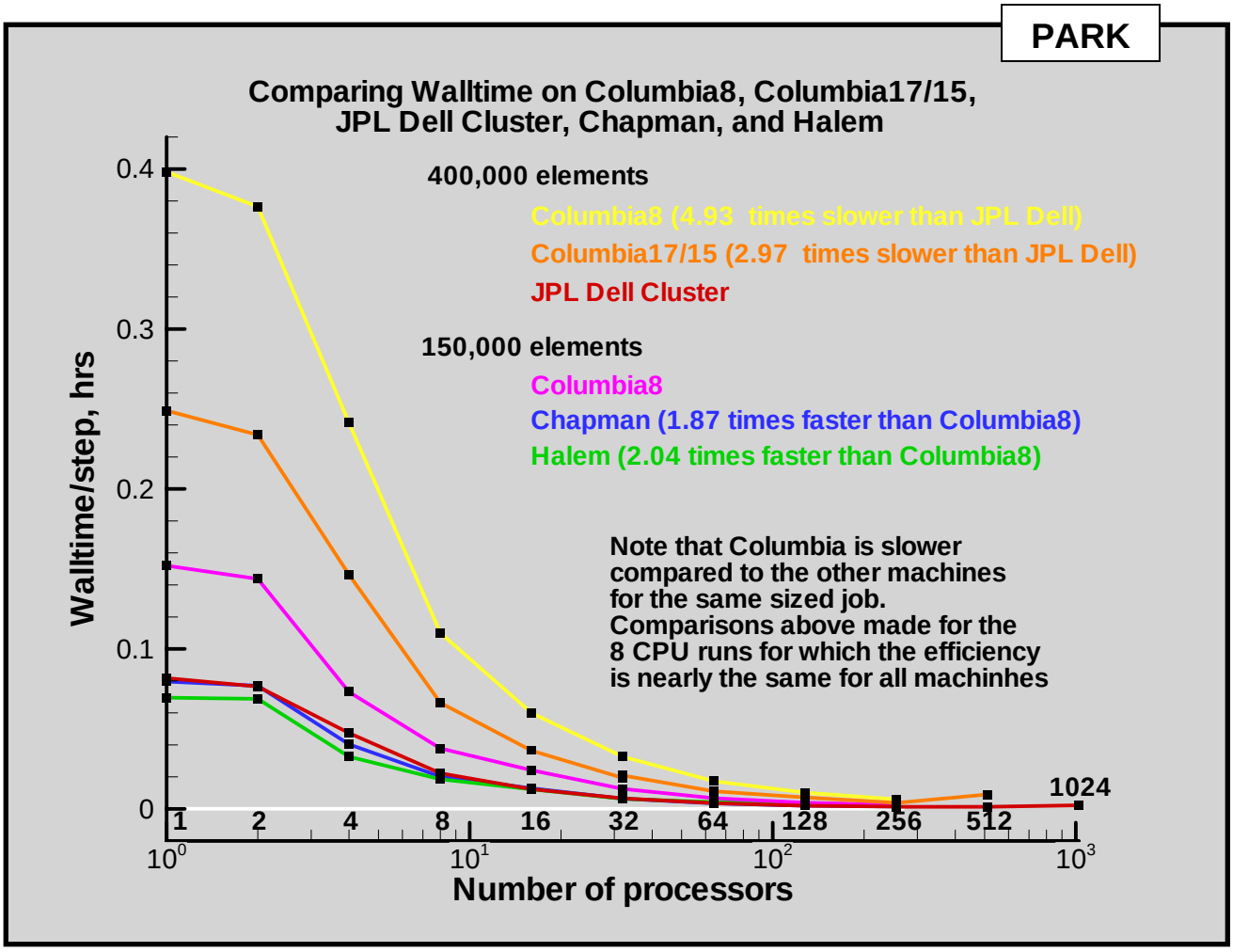


Figure 1 PARK Walltimes.

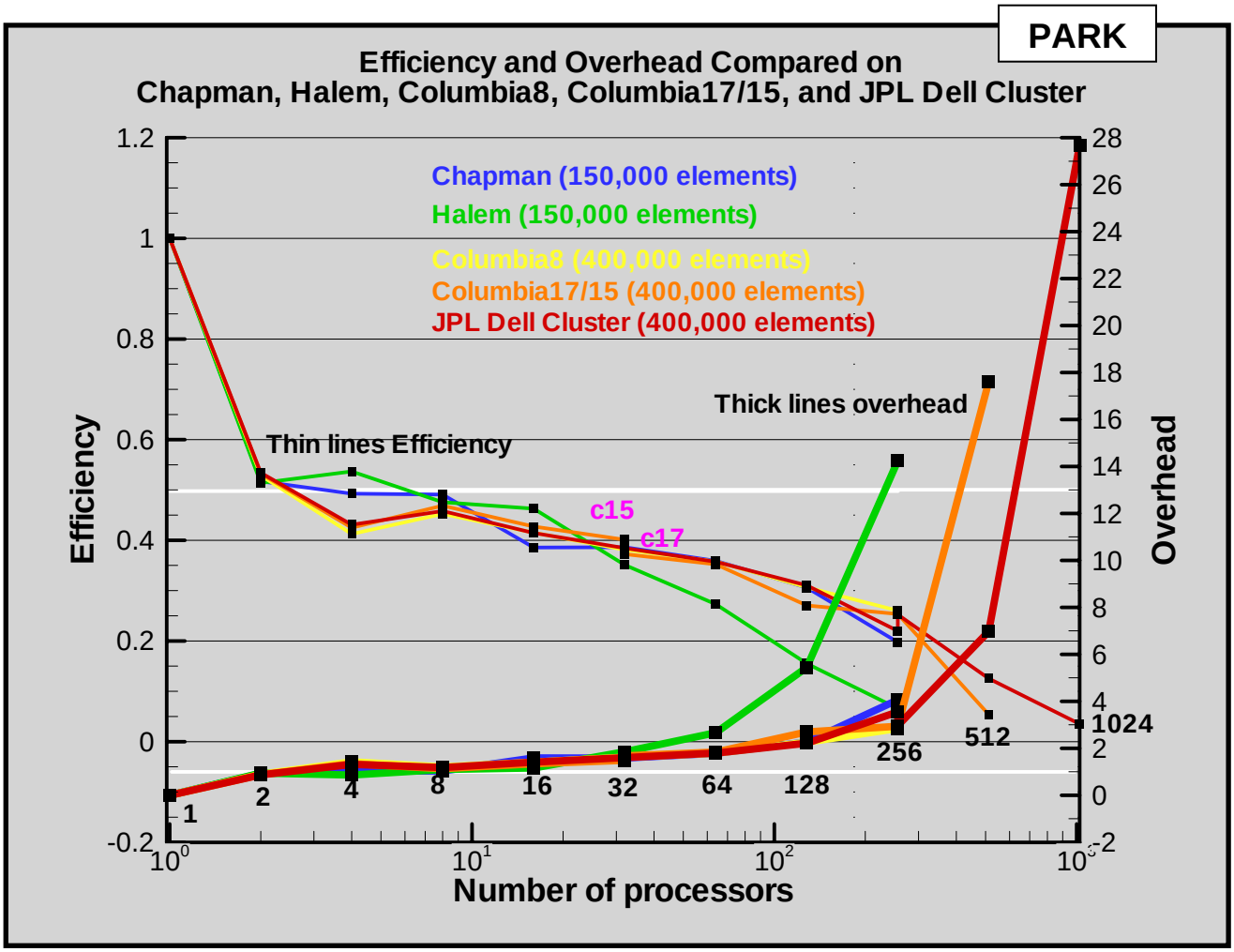


Figure 2: PARK: Efficiency and overhead comparison.

PARK Scientific and Computational Significance

Achieving the Second Code Improvement Milestone is significant because it opens the way to run significant sized problems since the earthquake code as well as the Fast Multipole library run in parallel under MPI. It presents to the scientific community fast parallel codes that allow creating simulations of the entire earthquake cycle in a 3D model that uses the most accurate description of fault friction, rate and state friction, and the quasi-dynamic radiation damping approximation to full elastodynamics. We have shown that the code can be run using a very large number of elements in the model compared to what could be done in the past.

This means that enough elements can now be used that is it possible to represent a reasonably sized fault with elements that are small enough that they can properly represent the behavior of a continuum. Larger numbers of elements also allow occurrence in the simulation of earthquakes with a large range of sizes. This means that it will be possible to study in the simulations in what situations small earthquakes occur in isolation and in what situations they may cascade or grow into larger ones. This could help gain an understanding of whether patterns of microseismicity might be used to help predict earthquakes. The attainment of this milestone not only represents an advance in our computational ability to simulate earthquakes, it will allow us to understand the earthquake process better by creating data sets that can be compared with data on real earthquakes.

PARK Simulation details

Code and documentation can be found at the web site

http://www.servogrid.org/slide/GEM/PARK/.

Within the appropriately named subdirectories under the

2nd_Code_Improv_Milestone directory in which this file is found can be found all the necessary material that describes the Second Code Improvement Milestone and gives instructions that would allow one to duplicate it. Included in the "in" and "out" directories are all the materials from the Second Code Improvement Milestone run with 400000 elements and 512 processors for 50,000 time steps. For code testing purposes on one's own system it is useful to set the number of time steps in the prk.dat.400003 file to a smaller number than 50000 for the initial run; even 1 or 2 would be reasonable for the first run.

The materials in these directories include:

Milestone_Certification_Data.txt - a file that gives the time required for the Second Code Improvement run and describes various parameters of the run.

README-setting_up_input_files.txt - a file that tells one how to understand the input files including an explanation of how the elements are created from the input files.

README-Compile.txt - a file that tells how to create both the multipole library and the PARK fault files using the appropriate Makefiles.

in - a directory that contains the input files that were used in the Second Code Improvement run.

out - a directory that contains the output files that were generated in the Second Code Improvement run.

src-bin - a directory that contains the PARK and related fault application files used in the Second Code Improvement run.

scaling - a directory that contains data and plots showing how execution time depends on number of elelents and processors. One file is a table giving the walltime per step for many scaling runs. One file is a Microsoft Word file with 2 imbedded plots that show the dependence on number of processors of

1) execution time
2) efficiency and
3) overhead.

In addition these 2 plots are also contained, one each, in 2 separate WindowsMetafile files.

downloads - a directory that contains a unix-compressed tar file, PARK_Package_2nd_Improv.tar.Z, that allows one to generate the files needed for the Second Code Improvement run. This compressed file is the only place on the website where all the files for the Fast Multipole Library can be found, since these are too large for convenient storage or downloading in their uncompressed form. All of these library files are in the t17-7 directory. If one wants to use the copyrighted Numerical Recipes routines that use the fifth order Runge Kutta scheme, please see either the README-src-bin.txt file in the src-bin directory or the header for the park.f file to learn what needs to be done to create the Numerical Recipes subroutines.

t17-7 - A directory containing the Fast Multipole Library. This is not explicitly included in the directory structure on the web site, but is in the directory structure that will be created when the compressed tar file is obtained, uncompressed and extracted.

The tar files were created in the following way.

While in the directory containing the 2nd_Code_Improv_Milestone directory, the following command was issued:

tar -cvf PARK_Package_2nd_Improv.tar 2nd_Code_Improv_Milestone

Then the .tar files were compressed by issuing:

compress PARK_Package_2nd_Improv.tar

to produce the PARK_Package_2nd_Improv.tar.Z file.

These can be uncompressed and the directory str4ucture and files restored on unix systems by issuing:

uncompress PARK_Package_2nd_Improv.tar.Z

and then

tar -xvf PARK_Package_2nd_Improv.tar

The Second Code Improvement Milestone run was as follows:

It had 50000 time steps and 400000 elements. It was run using 512 processors of the Dell Linux cluster at JPL and finished on Fri May 27, 2005 at 22:45:39. It was run by Charles Norton of JPL since Terry Tullis did not have access to this machine and Charles had done all the work to get the code running on the Dell cluster.

The execution time for the job can be seen from the output file:

prk.clocktime.400003

that can be found in the "out" directory that is contained in the directory in which this certification file resides on the website. The content of this file is copied here below

---

Thu May 26 12:14:33 2005

1 7.215867116311745E-013 3.607933558155872E-013

Thu May 26 12:14:52 2005

50000 1.904683118783539E-002 9.626188434109848E-007

Fri May 27 22:45:39 2005

Fri May 27 22:45:39 2005

The execution time can be found by taking the difference between the last time in this file, which is reported just before program termination, and the first time in this file, which is reported just after the program begins.

This difference is 34.5183 hours or 34h1m06s. This elapsed time has been shown from earlier tests to be essentially identical to the elapsed time one obtains by using the unix "time" command.

As required for this Second Code Improvement Milestone G, this time is less than 5 times that for the Baseline Milestone run. For comparison with the time for this Second Code Improvement Milestone G, The Baseline Milestone E run took 7h53m8.23s of real time according to the "time" command, or 7.8897 hours. Five times this is 39.4486 hours or 39h26m55s, greater than the Milestone G run.

PARK References

Salmon, John K, and Michael S. Warren, Parallel out-of-core methods for N-body simulation. In Michael Heath, Virginia, Torczon. et. al., editiors, *Eighth SIAM Conference on Parallel Processing for Scientific Computing,* SIAM, 1997.

Michael, S. Warren, John K. Salmon, Donald J. Becker, M. Patrick Goda, Thomas Sterling, and Gregoire S. Winckelmas. PentiumPro Inside: I. a treecode at 430 Gflops on ASCI red, II. Price/performance of $50/Mflop on Loki and Hyglac. In *Supercomputing, '97*, Los Alamos, 1997, IEEE Comp. Soc.